基于深度卷积神经网络的车辆检测

*初步设想：用faster rcnn模型，根据我自己的车辆检测数据集微调一下，观察检测效果。*

# Faster rcnn的测试步骤

程序：faster\_rcnn\_1/experiments/script\_faster\_rcnn\_demo.m;

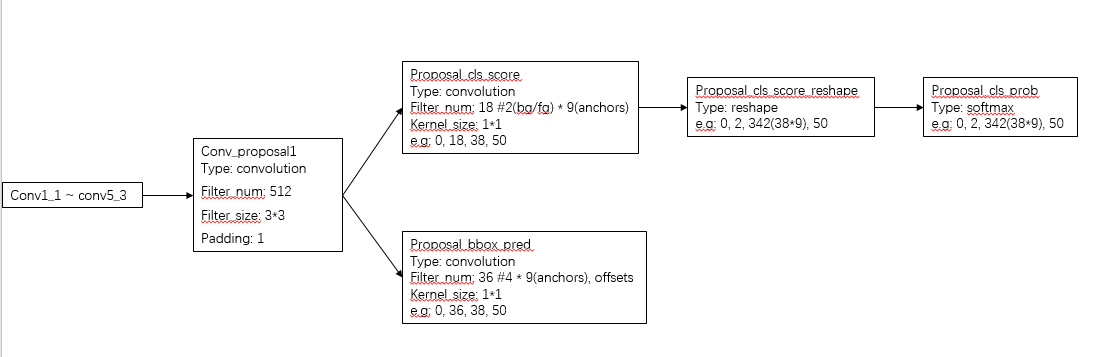
输入文件：faster\_rcnn\_1/output/\*;

输出结果：faster\_rcnn\_1/output/1.mat, 2.mat, 3.mat, 4.mat, 5.mat;

# 网络结构设计及测试思路

负责目标检测的网络有两个，分别是RPN (proposal\_test.prototxt)和Fast RCNN (detection\_test.prototxt)。二者采用共享卷积层的思想，因此共享卷积层仅在RPN中进行了定义，在Fast RCNN中仅从RoI pooling开始，网络的输入为boxes和feature maps。

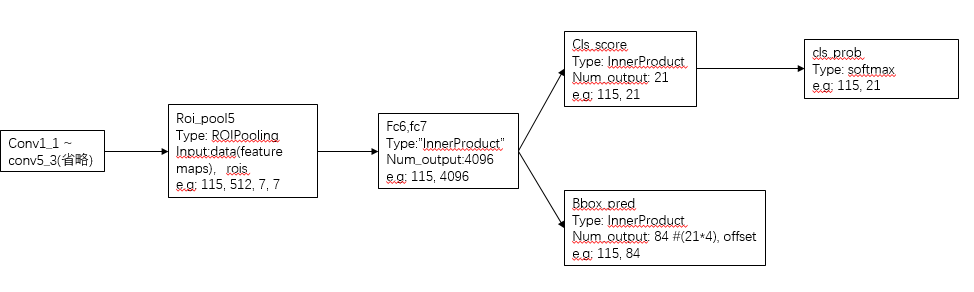
1. RPN网络：



测试思路：

通过RPN网络，得到两个结果，第一个是从图片中提取出的region相对anchor boxes的 offset，第二个是每个region的前景/背景概率。接下来，加载anchor boxes，与RPN输出的偏移进行运算，得到predicted boxes。将predicted regions按照置信度从高到低排序，然后进行两步筛选，第一步将region的数量与per\_nms\_topN进行比较取小，第二步对region进行NMS操作，去掉覆盖率超过nms\_overlap\_thres的区域，然后与after\_nms\_topN比较取小。结果保存在aboxes中（如115\*5）。

Fast RCNN网络：



测试思路：

由于在上一步中已经将较优的predicted regions筛选出来了，因此在输入Fast RCNN时，只要regions的四个参数，不需置信度。将predicted regions和共享卷积层的输出feature maps一起作为Fast RCNN的输入，Fast RCNN从RoI pooling layer开始定义，输出两个结果，第一个是class-specific BB的偏移（115\*84），第二个是分类概率（115\*21）。根据偏移和anchor boxes得到调整好位置的boxes，最后将背景相关的信息去除，boxes为前4列，scores为第一列。因此最终的输出结果就是boxes（115\*80），scores（115\*20）。

# 结果可视化

1. Object\_detection\_visualization.m

文件位置：faster\_rcnn\_1/experiments/object\_detection\_visualization.m

运行环境：服务器中

结果输出：faster\_rcnn\_1/output/1~5\_visual.mat

设计到nms算法，因此需要在服务器下运行。主要作用有两个，第一个作用是整理object\_detection结果，将boxes和scores按照类别整理成cell结构，每个cell对应一个类别，内容为regions\*5，包括每个区域的4个参数和该类别的概率。第二个作用是两步筛选，每个类别得到最具代表性的regions。第一步是用NMS去掉覆盖率超过0.3的regions；第二步是去掉预测概率低于class\_thres的regions。

1. Object\_detection\_getshowboxes.m, showboxes.m

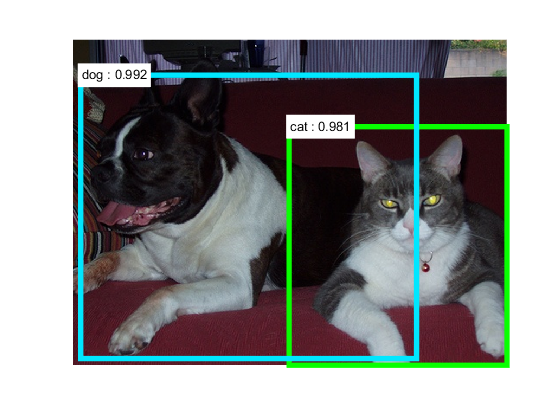
文件位置：D/

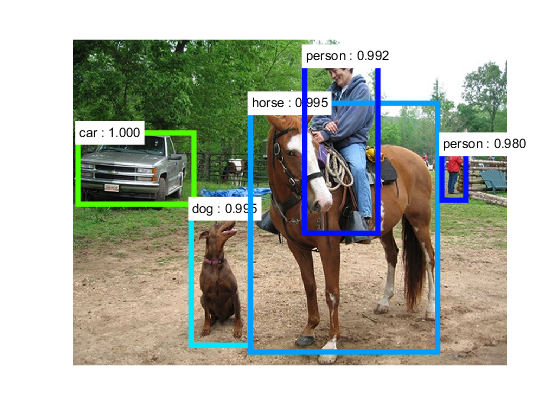
运行环境：本机的matlab上，matlab的工作目录为D：

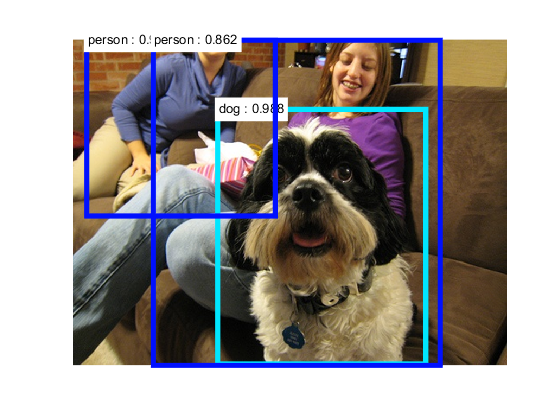
要画图，因此在本机的matlab上运行，object\_detection\_getshowboxes.m负责读取上一步的结果文件，并读取图片，调用showboxes.m画图。而showboxes.m是faster\_rcnn自带的程序，在faster\_rcnn\_1/utils/里，我只是把它复制到D盘下。

运行结果：D:/faster\_rcnn\_final\_model/1~5.bmp

部分结果如下图所示：







# Faster RCNN的微调阶段

程序文件：faster\_rcnn\_1/experiments/script\_faster\_rcnn\_VOC2007\_ZF.m；

修改后的文件：faster\_rcnn\_1/experiments/script\_faster\_rcnn\_Veh2017\_ZF.m；

功能：训练ZF\_5网络模型；

运行方式：在faster\_rcnn\_1目录下运行命令’sh train.sh’，数据库和网络模型都在程序中固定，不需传入参数。数据库路径设定在./experiments/+Dataset/private/veh2017\_devkit.m中，数据库名称，目标类别等设定在./datasets/VOCdevkit2007/VOCcode/VOCinit.m中；

网络训练结果保存在faster\_rcnn\_1/output/faster\_rcnn\_final/faster\_rcnn\_Veh2017\_ZF/中。

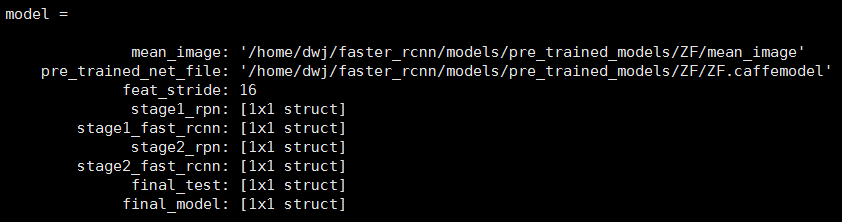
## Faster RCNN数据处理

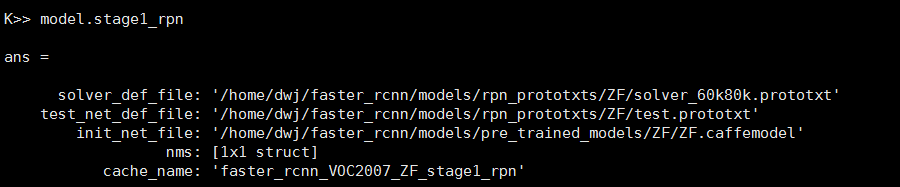
Faster RCNN采用的数据集是VOC2007，而我需要自己制作数据集，因此要了解Faster RCNN需要数据集的哪些内容。

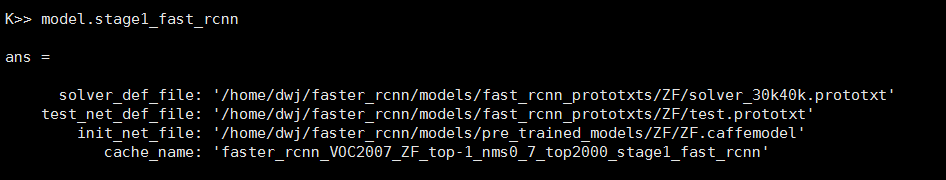
我的第一个目标是使函数可以成功加载数据集。在官网中会下三个文件，分别是[training/validation data](http://host.robots.ox.ac.uk:8080/pascal/VOC/voc2007/VOCtrainval_06-Nov-2007.tar)，[development kit code and documentation](http://host.robots.ox.ac.uk:8080/pascal/VOC/voc2007/VOCdevkit_08-Jun-2007.tar)，[annotated test data](http://host.robots.ox.ac.uk:8080/pascal/VOC/voc2007/VOCtest_06-Nov-2007.tar)。

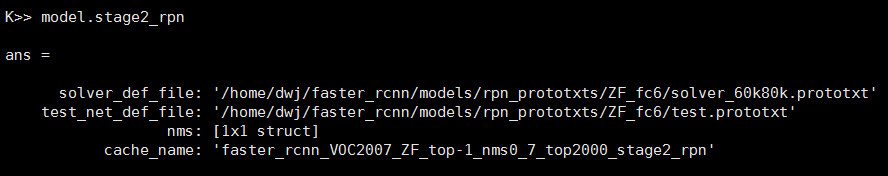
创建目录faster\_rcnn/datasets/VOCdevkit2007/，然后将*development*中的内容全部拷到该文件夹下；创建目录faster\_rcnn/datasets/VOCdevkit2007/VOC2007/，将*traval,test*里Annotations, ImageSets, JPEGImages三个文件夹中的数据合并后拷到该目录下。数据安置完毕。

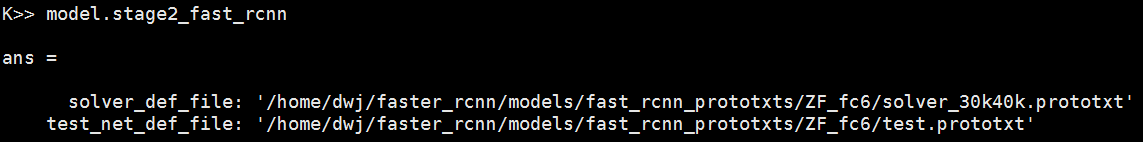
该程序的第一步是加载模型ZF模型，指定图片平均值，预训练权重，四步走训练每一步的参数，以及最后测试时的region proposal数量阈值等：

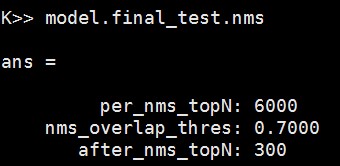












上述网络结构和参数的设定是固定的，不需要更改，需要更改的是后边两条语句：

dataset = Dataset.voc2007\_trainval(dataset, ‘train’, use\_flipped);

dataset = Dataset.voc2007\_test(dataset, ‘test’, false);

1. Dataset.voc2007\_trainval()相关信息：

* 调用imdb\_from\_voc(devkit, ‘trainval’, ‘2007’, use\_flip) 时涉及到的信息：

datasets/VOCdevkit2007/VOC2007/Annotation：全都是.xml，标注对应图片的标注；

datasets/VOCdevkit2007/VOC2007/JPEGImages：.jpg；

datasets/VOCdevkit2007/VOC2007/ImageSet/Main %s.txt：train，trainval，val，test数据集的所有图片名称列表；

datasets/VOCdevkit2007/VOC2007/ImageSet/Main %s\_%s.txt：所有类别的.txt文件，如bicycle\_train.txt表示train数据集中所有图片是否包含该类别的目标，有就为1，没有就为-1；

datasets/VOCdevkit2007/results/VOC2007/Main/%s\_cls\_val\_%s.txt：结果文件，不用管；

datasets/VOCdevkit2007/results/VOC2007/Main/%s\_det\_val\_%s.txt：结果文件，不用管；

class：{ 'bus' , 'car' , van' , 'person' }；(这是我自己数据库的类别)

后边的都是后来生成的，不用管。

在读数据方面，读了./datasets/VOCdevkit2007/VOC2007/JPEGImages/ 中的图片；读了./datasets/VOCdevkit2007/VOC2007/ImageSets/Main/trainval.txt数据。

结果存储在./imdb/cache/imdb\_voc\_2007\_trainval\_flip.mat中

* 调用roidb\_from\_voc(devkit, ‘trainval’, ‘2007’, use\_flip) 时涉及到的信息：

用到了./datasets/VOCdevkit2007/VOC2007/Annotations/里边的.xml数据作为区域提名的groundtruth。调用的函数为PASreadrecord()，所在位置为./datasets/VOCdevkit2007/VOCcode。

结果存储在./imdb/cache/roidb\_voc\_2007\_trainval\_flip\_easy.mat中。

1. Dataset.voc2007\_trainval()相关信息：

* 调用imdb\_from\_voc(devkit, ‘test’, ‘2007’,false) 时涉及到的信息：

大体同trainval，在读数据方面，读了./datasets/VOCdevkit2007/VOC2007/JPEGImages/ 中的图片；读了./datasets/VOCdevkit2007/VOC2007/ImageSets/Main/test.txt数据。

结果存储在./imdb/cache/imdb\_voc\_2007\_test.mat中。

* 调用roidb\_from\_voc(devkit, ‘test’, ‘2007’, false) 时涉及到的信息：

大体同trainval，结果存储在roidb\_voc\_2007\_test\_easy.mat中。

1. 总结：

dataset构造完毕之后，需要更改的函数文件：

VOCinit\_1.m: datasets/VOCdevkit2007/VOCcode/VOCinit.m(VOCinit\_1.m)

veh2017\_devkit.m: ./experiments/+Dataset/private/veh2017\_devkit.m(voc2007\_devkit.m)

veh2017\_trainval.m: ./experiments/+Dataset/veh2017\_trainval.m(voc2007\_trainval.m)

imdb\_from\_veh.m: ./imdb/imdb\_from\_veh.m(imdb\_from\_voc.m)

veh2017\_test.m: ./experiments/+Dataset/veh2017\_test.m(voc2007\_test.m)

roidb\_from\_veh.m: ./imdb/roidb\_from\_veh.m(roidb\_from\_voc.m)

在后边测试求解mAP的时候，do\_fast\_rcnn\_test() → fast\_rcnn\_test() → imdb\_eval\_voc()时，imdb\_eval\_voc也需要更改，更改内容包括year的获取位置，以及do\_val的判断，目的是使do\_val为1。

imdb\_eval\_voc.m(): ./imdb/imdb\_eval\_voc.m

不需更改的函数文件：

PASreadrecord.m: ./datasets/VOCdevkit2007/VOCcode/PASreadrecord.m

## 构造Faster RCNN数据

1. 训练和测试图片准备

我在网上没有找到专门的数据库，现有的数据库都是每一个图片仅包含一辆车，与我的车辆检测任务不符，最后决定自己录一些车流的视频，再转成图片。下面是测试视频转图片的方法：.avi → .jpg

工具：ffmpeg；

平台：linux服务器，已经安装了ffmpeg；

操作：在/home/dwj/video下，运行man ffmpeg | grep image，查看ffmpeg与图片处理相关的操作。视频转图片的指导如下：



将foo.avi替换成自己的视频文件，将W\*H替换成输出图片的规模，4032\*3024，foo-%03d。jpeg是输出图片的名称和格式，可以改为.img/%06d.jpg，因为faster rcnn的数据全是6位命名格式。完整命令如下所示：



进入img文件夹下就可看到转化成功的.jpg图片，采用显著性帧提取，因此图片的数量是不需指定的。但是每张图片的重复度很高，只是每辆车发生了位移，后续还会再研究他的显著帧提取相关参数，尽量做到每张图片都有新的内容呈现。

1. 画目标包围框

手动标注ground truth，将图片中所有的目标用矩形框框起来，并注上类别，结果保存为txt文件，为后边转xml做准备。

工具：图片标注工具v2；

平台：windows；

操作：运行media.exe，结果保存在\*.txt中，每个图片为一行，格式为：

图片路径 目标数量 五元组（boxes四个参数，类别序号）…

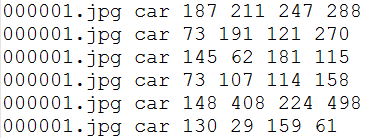


但是这种txt格式不符合.xml转换的输入标准，还需要有一个txt的格式转换工具。

工具：ChangeTxtFormat.m；

平台：matlab；

操作：将上一步生成的\*.txt保存在matlab工作目录下，如D:\，并在该目录下运行ChangeTxtFormat.m，生成结果文件output.txt。格式为：



1. 制作annotations文件，同时生成ImageSets\Main\train.txt，trainval.txt，val.txt，test.txt

工具：VOC2007xml.m；

平台：matlab;

操作：一次性处理一个数据库，将上一步生成的output.txt文件，图片，保存在VOC2007xml\img\下，运行程序之后，\*.xml会保存到VOC2007xml\Annotations\下，图片也会被copy到VOC2007xml\JPEGIMages\下。运行方式是将VOC2007xml这个文件夹拷贝到matlab的工作目录下，如D:\，接着运行VOC2007xml.m即可。

生成的xml文件与VOC2007标准的xml文件格式是一致的，但是涉及到<truncated>节点和<difficult>节点时，默认都为0，可能影响不大。在生成\*.xml的同时，也在VOC2007xml\ImageSets\Main\下的train(test…).txt中写出一条图片的编号。

1. 更改Faster RCNN的相关文件夹和文件

* 首先重命名datasets中的文件：

./datasets/VOCdevkit2007 → ./datasets/Vehicledevkit2017；

./datasets/VOCdevkit2007/VOC2007 → ./datasets/Vehicledevkit2017/Vehicle2017；

./datasets/VOCdevkit2007/local/VOC2007 → ./datasets/Vehicledevkit2017/local/Vehicle2017；

./datasets/VOCdevkit2007/results/VOC2007 → ./datasets/Vehicledevkit2017/results/Vehicle2017；

* 上传文件

VOCinit.m：./datasets/Vehicledevkit2017/VOCcood/;

veh2017\_devkit.m: ./experiments/+Dataset/private/veh2017\_devkit.m(voc2007\_devkit.m);

veh2017\_trainval.m: ./experiments/+Dataset/veh2017\_trainval.m(voc2007\_trainval.m);

imdb\_from\_veh.m: ./imdb/imdb\_from\_veh.m(imdb\_from\_voc.m);

veh2017\_test.m: ./experiments/+Dataset/veh2017\_test.m(voc2007\_test.m);

roidb\_from\_veh.m: ./imdb/roidb\_from\_veh.m(roidb\_from\_voc.m)

* 上传数据

./datasets/Vehicledevkit2017/Vehicle2017/目录下有三个文件，分别是Annotations，ImageSets，JPEGImages，前三步的准备已经准备了这些文件，直接替换即可。目前，在ImageSets/Main中只用到了两个文件，分别是trainval.txt和test.txt，所以先更改这两个文件并上传就好。

* 检查修改结果

入手文件./experiments/script\_faster\_rcnn\_Veh2017\_ZF.m(script\_faster\_rcnn\_VOC2007\_ZF.m)

运行程序之后在./imdb/cache/目录下有四个文件，为数据信息和region proposal信息。

imdb\_veh\_2017\_trainval\_flip.mat;

roidb\_veh\_2017\_trainval\_flip\_easy.mat;

imdb\_veh\_2017\_test.mat;

roidb\_veh\_2017\_test\_easy.mat;

1. 批量数据制作：

* 录制视频（一分钟一个）；
* 在linux服务器上，将视频传输到/home/dwj/video/文件夹下，用命令提取图片：

ffmpeg –I \*.avi -r 1 -s 4032x3024 -f image2 ./img/%06d.jpg，结果保存在/home/dwj/video/img文件夹下。将img文件夹下载到D:\下，清空img文件夹，为下一个视频做准备；

* 在matlab中用ResizeImage.m将D:\img文件夹下的图片resize成原来的0.2倍，否则图片标注工具会因内存不足而停止工作，依然写回到原文件夹内。
* 用画图软件辅助，必要时裁剪，筛选可以用来做标注的图片；
* 在matlab中用RegImageName.m将图片全部重命名，将文件的名称按照数字顺序重新排下来，注意要更改文件的起始序号。将重命名后的文件保存到D:\VOC2007xml\img文件夹内；
* 用标注工具标注图片，结果文件重命名为videoname.txt(0417.txt)，将文件剪切到D:\original\_label\目录下；
* 在matlab下运行ChangeTxtFormat.m将上一步的标注结果进行整理，每个目标占一行，得到输出D:\img\number\_output.txt(0417\_output.txt)，注意更改输出文件的名称；
* 将matlab的当前目录改为D:\VOC2007xml，运行VOC2007xml.m，注意更改输入文件的名称。生成图片的.xml文件，同时将图片剪切到JPEGImages文件夹里，另外在ImageSets中生成或追加trainval.txt文件。

## Faster RCNN四步走训练

Faster rcnn的四步走训练分别起名为：

stage1\_rpn.cache\_name: faster\_rcnn\_Veh2017\_ZF\_stage1\_rpn;

stage1\_fast\_rcnn.cache\_name: faster\_rcnn\_Veh2017\_ZF\_top-1\_nms0\_7\_top2000\_stage1\_fast\_rcnn;

stage2\_rpn.cache\_name: faster\_rcnn\_Veh2017\_ZF\_top-1\_nms0\_7\_top2000\_stage2\_rpn;

stage2\_fast\_rcnn.cache\_name: faster\_rcnn\_Veh2017\_ZF\_top-1\_nms0\_7\_top2000\_stage2\_fast\_rcnn;

1. Stage1\_rpn训练：

Anchors的生成与训练图片无关，是根据参数生成的，保存在./output/rpn\_cachedir/faster\_rcnn\_Veh2017\_ZF\_stage1\_rpn/anchors.mat中，九个anchors。

训练数据库：trainval;

评估数据库：test;

权重文件：./models/pre\_trained\_models/ZF/ZF.caffemodel;

参数定义文件：./models/rpn\_prototxts/ZF/solver\_60k80k.prototxt;

网络结构定义文件：./models/rpn\_prototxts/ZF/train\_val.prototxt;

微调后的权重（是个文件）：

./output/rpn\_cachedir/faster\_rcnn\_Veh2017\_ZF\_stage1\_rpn/veh\_2017\_trainval/final

1. Stage1\_rpn测试：

Conf：conf\_proposal；

第一步：Imdb：dataset.imdb\_train；

第二步：imdb：dataset.imdb\_test；

网络结构定义文件(net\_def\_file)：./models/rpn\_prototxts/ZF/test.prototxt；

权重文件(net\_file)：

./output/rpn\_cachedir/faster\_rcnn\_Veh2017\_ZF\_stage1\_rpn/veh\_2017\_trainval/final；

Cache\_name：faster\_rcnn\_Veh2017\_ZF\_stage1\_rpn

思路是用RPN网络预测出predicted boxes的偏移和，fg/bg概率。根据anchor boxes与偏移的联合作用得到predicted boxes的4个绝对参数，以fg的概率作为置信度。根据置信度（score）的降序排列重新排序score和predicted boxes的顺序。经过NMS 处理得到最终结果。

rpn\_test分两步进行，第一步求roidb\_train；第二步求roidb\_test，下述保存的结果只是做到了去除某些小型boxes以及按scores降序排列，NMS等后续处理要在do\_proposal\_test.m的后续语句中进行，结果赋给dataset.roidb\_train和dataset.roidb\_test。

用RPN网络预测出的boxes存储位置：

./output/rpn\_cachedir/faster\_rcnn\_Veh2017\_ZF\_stage1\_rpn/veh\_2017\_trainval/proposal\_boxes\_veh\_2017\_trainval.mat

./output/rpn\_cachedir/faster\_rcnn\_Veh2017\_ZF\_stage1\_rpn/veh\_2017\_test/proposal\_boxes\_veh\_2017\_test.mat

1. Stage1\_fast\_rcnn训练：

在前两步，RPN和Fast\_rcnn单独训练，所以不共享权重，初始化权重全部为预训练权重。但是他会用到第一步的region proposal，与第一步RPN保存在相同的目录中。

训练数据库：trainval;

评估数据库：test;

权重文件：./models/pre\_trained\_models/ZF/ZF.caffemodel;

参数定义文件：models/fast\_rcnn\_prototxts/ZF/solver\_30k40k.prototxt;

网络结构定义文件：./models/ fast\_rcnn\_prototxts /ZF/train\_val.prototxt;

微调后的权重（是个文件）：

./output/fast\_rcnn\_cachedir/faster\_rcnn\_Veh2017\_ZF\_top-1\_nms0\_7\_top2000\_stage1\_fast\_rcnn/veh\_2017\_trainval/final

1. Stage1\_fast\_rcnn测试—求mAP

测试数据库为test;

权重文件：

./output/fast\_rcnn\_cachedir/faster\_rcnn\_Veh2017\_ZF\_top-1\_nms0\_7\_top2000\_stage1\_fast\_rcnn/veh\_2017\_trainval/final

网络结构定义文件：./models/fast\_rcnn\_prototxts/ZF/test.prototxt

结果存放目录：

./output/fast\_rcnn\_cachedir/faster\_rcnn\_Veh2017\_ZF\_top-1\_nms0\_7\_top2000\_stage1\_fast\_rcnn/veh\_2017\_test

1. Stage2\_rpn训练：

从第三步起，RPN和fast rcnn要共享卷积层，所以它的初始化权重文件为上一步fast rcnn的微调结果，而且仅微调非共享层的权重。

权重文件：

./output/fast\_rcnn\_cachedir/faster\_rcnn\_Veh2017\_ZF\_top-1\_nms0\_7\_top2000\_stage1\_fast\_rcnn/veh\_2017\_trainval/final

参数定义文件：./models/rpn\_prototxts/ZF/solver\_60k80k.prototxt;

网络结构定义文件：./models/ fast\_rcnn\_prototxts /ZF/train\_val.prototxt;

微调后的权重（是个文件）：

output/rpn\_cachedir/faster\_rcnn\_Veh2017\_ZF\_top-1\_nms0\_7\_top2000\_stage2\_rpn/veh\_2017\_trainval/final

1. Stage2\_rpn测试：

Conf：conf\_proposal；

第一步：Imdb：dataset.imdb\_train；

第二步：imdb：dataset.imdb\_test；

网络结构定义文件：./models/ rpn\_prototxts /ZF\_fc6/test.prototxt;

权重文件：

./output/rpn\_cachedir/faster\_rcnn\_Veh2017\_ZF\_top-1\_nms0\_7\_top2000\_stage2\_rpn/veh\_2017\_trainval/final

用RPN网络预测出的boxes存储位置：

./output/rpn\_cachedir/faster\_rcnn\_Veh2017\_ZF\_top-1\_nms0\_7\_top2000\_stage2\_rpn/veh\_2017\_trainval/proposal\_boxes\_veh\_2017\_trainval.mat

./output/rpn\_cachedir/faster\_rcnn\_Veh2017\_ZF\_top-1\_nms0\_7\_top2000\_stage2\_rpn/veh\_2017\_test/proposal\_boxes\_veh\_2017\_test.mat

1. Stage2\_fast\_rcnn训练：

与RPN共享卷积层的权重，所以初始化权重也是上一步fast rcnn的微调结果，仅微调非共享层的权重。

权重文件：

./output/fast\_rcnn\_cachedir/faster\_rcnn\_Veh2017\_ZF\_top-1\_nms0\_7\_top2000\_stage1\_fast\_rcnn/veh\_2017\_trainval/final

参数定义文件：models/fast\_rcnn\_prototxts/ZF\_fc6/solver\_30k40k.prototxt;

网络结构定义文件：./models/ fast\_rcnn\_prototxts /ZF\_fc6/train\_val.prototxt;

微调后的权重（是个文件）：

./output/fast\_rcnn\_cachedir/faster\_rcnn\_Veh2017\_ZF\_top-1\_nms0\_7\_top2000\_stage2\_fast\_rcnn/veh\_2017\_trainval/final

1. 最后的测试

对test数据库提取RoI，网络结果定义为./models/ rpn\_prototxts /ZF\_fc6/test.prototxt，网络权重为RPN第二个阶段的训练结果权重，然后进行test的mAP求解。

求解过程中回生成三种文件：

* Class\_boxes\_veh\_2017\_test.mat：保存所有测试图片的所有符合要求的predicted boxes五个参数输出，其中第五个参数是对应class的概率。本次测试包含250张测试图片。Class包括bus, car, van, person四类，因此有四个这样的文件。
* Class\_pr\_veh\_2017\_test.mat：某类别的准确率召回率信息；
* Class\_pr\_veh\_2017\_test.jpg：某类别的准确率召回率曲线；

文件保存位置：

./output/fast\_rcnn\_cachedir/faster\_rcnn\_Veh2017\_ZF\_top-1\_nms0\_7\_top2000\_stage2\_fast\_rcnn

检测结果：

|  |  |
| --- | --- |
|  | AP(%) |
| bus | 100 |
| car | 90.8935 |
| van | 58.7413 |
| person | 59.9736 |
| mAP | 77.4021 |

1. 结果融合：

结果存放目录：./output/faster\_rcnn\_final/faster\_rcnn\_Veh2017\_ZF

存放内容：

* Proposal\_test.prototxt：RPN的网络结构，原models/fast\_rcnn\_prototxts/ZF\_fc6/test.prototxt;
* Proposal.final: RPN权重，RPN stage2的微调结果；
* Detection\_test.prototxt: fast rcnn的网络结构，在原models/fast\_rcnn\_prototxts/ZF\_fc6/test.prototxt的基础上去掉共享层，从pooling5 开始；
* Detection\_final: fast rcnn权重，fast rcnn stage2的微调结果；

注意：程序不会自动去掉fast rcnn中的共享层，需要手动去除。

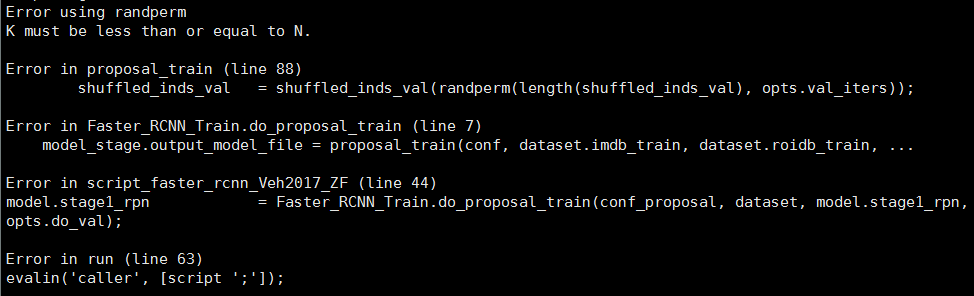
## 将自己的数据送入Faster RCNN进行训练

1. 更改平均值文件

原数据的平均值文件存储在./model/pre\_trained\_models/ZF/mean\_image.mat中，是一个224\*224\*3的三维矩阵。由于我数据库与之不同，所以要重新计算平均值，每个通道只计算一个平均值，为了配合它的格式，也要将平均值做成三维矩阵的形式，只是每个通道的所有值相同。求平均值的思路是将所有图片对应通道的像素值相加后整体求平均，程序文件为GetImageMean.m

1. 修改stage1\_rpn和stage1\_fast\_rcnn的训练参数

如果评估数据不够，在训练过程中会出现错误：



因此要将./functions/rpn/proposal\_train.m进行更改，将语句



改为：



将./functions/fast\_rcnn/fast\_rcnn\_train.m的同样语句进行更改：



1. 在做stage1\_fast\_rcnn训练的时候，更改models\fast\_rcnn\_prototxts\ZF\train\_val.prototxt和test.prototxt这两个网络结构定义文件。其中cls\_score层的输出改为5(4个类别+背景)，bbox\_pred的输出改为20（5个类别输出×4个参数）。同时要更改文件开头处的”bbox\_tragets”和“bbox\_loss\_weights”中的维度，将84改为20。
2. 在做stage2\_fast\_rcnn训练的时候，更改models\fast\_rcnn\_prototxts\ZF\_fc6\train\_val.prototxt和test.prototxt这两个网络结构定义文件。其中cls\_score层的输出改为5(4个类别+背景)，bbox\_pred的输出改为20（5个类别输出×4个参数）。同时要更改文件开头处的”bbox\_tragets”和“bbox\_loss\_weights”中的维度，将84改为20。

## 将测试集的测试结果进行可视化输出

在网络完全训练完毕之后，会得到一个final model，存储在./output/faster\_rcnn\_final/faster\_rcnn\_Veh2017\_ZF/中，包含RPN和Fast RCNN的网络结构定义和网络权重，还有一个model.mat，记录平均值等信息。注意Fast RCNN的网络结构定义要手动删除与RPN网络重合的部分，也就是从conv1~relu5。还要改最开始的input’data’部分，该输入维度，输入的应该是relu5输出后的维度。

在./output下创建两个文件夹，250\_testimages(据具体的测试图片数量进行名称更改)，存放测试图片的目标检测结果；test\_visualization，存放可视化结果。

接下来，更改./experiment/script\_faster\_rcnn\_demo.m中的内容，更改后的文件为./experiment/script\_faster\_rcnn\_1\_demo.m，它实现的功能是对制定的图片做目标检测，并将可视化结果保存为.png格式文件，便于在本机电脑上查看。注意，script\_faster\_rcnn\_demo.m本身具有可视化的功能，只是服务器无法可视化，所以就更改程序将可视化结果保存后在本机上查看。另外也没有直接用训练过程中得到的region等结果，因为他们与script\_faster\_rcnn\_demo.m中要求的输入有偏差，所以统一重新计算。

修改文件如下，保存在Vehicle2017更改的程序/可视化结果程序/下：

1. ./experiments/script\_faster\_rcnn\_1\_demo.m

INIT\_MODEL部分：更改model的目录；

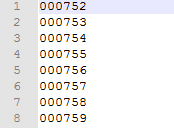
TESTING部分：更改测试文件集合，读取./datasets/Vehicledevkit2017/Vehicle2017/ImageSets/Main/test.txt中的所有图片名称，逐个进行可视化处理；在目标检测部分，所有检测结果都保存在目录./output/250\_testimages/下，先检查该图片是否经历了目标检测，若已经检测，就直接加载结果，结果包括region proposal和scores两种变量。将目标检测结果，图片，以及可视化结果名传入showboxes，进行可视化结果保存。

1. ./utils/showboxes.m

增加一个函数参数，image\_name，为可视化结果文件名；在程序的一开始加入语句figure(1)，便于后边print()抓取结果保存；更改颜色判断，将’voc’改为’veh’；在程序最后加入print()语句，将可视化结果保存为png文件，所有结果存储在目录./output/test\_visualization/下。

1. 运行方式

在faster\_rcnn\_1目录下输入命令’sh test.sh 图片目录 图片名称列表文件.txt’。图片名称列表文件是一个记录所有待检测的图片名称（不带后缀）的txt文件，如下所示：



结果存放在两个文件夹中，分别是250\_testimages（前缀根据真实文件数量改变），存放目标检测的坐标结果和分类结果；test\_visualization，存放可视化结果。

观察可视化结果，发现以下几点问题：

1. 测试结果中，不止一次将白色树干的某个部位认为是人，且分类概率高于60%。图片标号：753，775（虽然人在树干后，但几乎被完全挡住，看不出人类特征，所以认为是把树干当成了人，与753是同一树干，几乎在同一位置），781，787，790，812，813，817，822，825，826，827，838，847，850，851，852，855，856，860，861，862，869，870，871，873，882；

将其他位置错检为person，754，874（将垃圾桶视为人，垃圾桶是站立式内凹型垃圾桶，从轮廓来看确实像人，分类概率高于80%），834（车后屁股加车轮），888（垃圾车）；

1. Person类别在region proposal时几乎会将人和座骑一起框出，符合要求。
2. 在辨认van类别时，将van视为car，或将car视为van(也有可能是我自己标注的时候就不是很能区分这些)。图片标号：775，831，832；
3. 当人与车很接近时，有时检测不出人。图片标号：823，826，831，835，915，918，960，998；
4. 当人与背景颜色较接近时，检测不出人。图片编号：895，896，921；

但是在某个场景下，人和背景颜色差别较大时，仍无法检测出人，906，913，927，962，964，996，997；

把几个人框在一起，916，961，980，981，982，983

1. 极个别情况，连车都没有检测完全。图片标号：827，842